IN THE SPECIFICATION

Please amend the specification as follows:

The paragraph beginning at page 1, line 13 is amended as follows:

In the rapid development of computers many advancements have been seen in the areas of processor speed, throughput, communications, and fault tolerance. Initially, computer systems were standalone devices in which a processor, memory, and peripheral devices all communicated through a single bus. Later, in order to improve performance, several processors were interconnected to memory and peripherals using one or more buses. In addition, separate computer systems were linked together through different communications mechanisms such as, shared memory, serial and parallel ports, local area networks (LAN), and wide area networks (WAN). Further, in order to improve processor instruction processing, pipelining was developed to enable a processor to execute an instruction in stages and a single processor could execute different instructions at different stages of execution simultaneously.

The paragraph beginning at page 2, line 3 is amended as follows:

A further development created in order to enhance processor performance is the use of a technique known as single instruction multiple data (SIMD). SIMD is a technique where several different pieces of data may be simultaneously accessed and arithmetically manipulated by a processor. This ability to manipulate several pieces of data at the same time greatly enhances the performance of the processor. However, even though the same arithmetic operation may be

performed, the results and status for each piece of data may be different. For example, the data may be negative, zero, have a carry out or overflow condition resulting. Since a SIMD processor may manipulate as many as eight pieces, or more, of data simultaneously, the processor is required to maintain at least eight sets of these condition flags. Further, in order to receive the benefit of SIMD processing, it is necessary to logically combine these condition or arithmetic flags so that the appropriate operation may occur under the appropriate conditions. Since it may be necessary to manipulate eight pieces, or more, of data under many different combinations of possible outcomes, the logic that must be built into a processor and microprocessor design can be very cumbersome. Valuable space on the microprocessor must be dedicated to this processing and the speed, size, power required, and heat generated by the processor may be seriously ~~effected~~ affected.

The paragraph beginning at page 6, line 15 is amended as follows:

FIG. 2 is a systems diagram of an example embodiment of the present invention. As illustrated in FIG. 1B, arithmetic flags 120, 125, 130 and 135 are shown in FIG. 2. However, in addition, arithmetic flags 120, 125, 130 and 135 are each associated with data items 100, 105, 110 and 115 respectively. As previously discussed, in order for [[a]] an SIMD capable processor, such as processor 165, to effectively be able to manipulate multiple pieces of data (100 B 115) it is necessary to logically combine the results of mathematical operations shown in arithmetic flags 100, 125,130 and 135. This is accomplished by the combination function module 160 utilizing the methods and operations illustrated and further discussed in reference to FIGs. 3 B 6. The results of the combination function performed by the combination function

module 160 is a combined arithmetic flag variable 170. Thereafter, a condition check module 175 is utilized to determine the next operation to perform based upon the combined arithmetic flag variable 170. These operations will be discussed further detail ahead.

Please amend the paragraph beginning at page 7, line 15, as follows:

Before proceeding into a detailed discussion of the logic used by the present invention it should be mentioned that the flowcharts shown in FIGs. 3 through 6 ~~or contain~~ illustrate software, firmware, hardware, processes or operations that correspond, for example, to code, sections of code, instructions, commands, objects, hardware or the like, of a computer program that is embodied, for example, on a storage medium such as floppy disk, CD-Rom (Compact Disc read-only Memory), EP-Rom (Erasable Programmable read-only Memory), RAM (Random Access Memory), hard disk, etc. Further, the computer program can be written in any language such as, but not limited to, for example C ++. Further, the logic shown in figs 3 B 6 are executed by the modules and processor 165 shown in FIG. 2.

The paragraph beginning at page 8, line 1 is amended as follows:

FIG. 3 is an of an example flowchart of a general embodiment of the present invention. Logic utilized in the flowchart illustrated in FIG. 3 ~~maybe~~ may be used to combine, group, or extract the arithmetic flags illustrated in FIGs. 1A through 1B. The functions that may be executed by the condition check module 175 would include, but not be limited to, the following functions.

The paragraph beginning at page 8, line 22 is amended as follows:

As would be appreciated by one ~~order~~ of ordinary skill [[of]] in the art, the foregoing functions may be increased to include any mathematical functions including less than, greater than, less than or equal to, and greater than or equal to. Additional, mathematical operators and functions may be used in conjunction with the present invention.

The paragraph beginning at page 9, line 3 is amended as follows:

Still referring to FIG. 3, processing begins in operation 200 and immediately proceeds to operation 210. In operation 210, a field size is determined on which to base the extraction or combination function. The field size may be, but not limited to, a nibble, byte, half word, word, or double word in size. The extraction and/or combination function may include any of the foregoing 16 items discussed or any other function which may describe or combine the status or result of a mathematical operation performed by a computer or processor. Thereafter, processing proceeds to operation 220 where it is determined if an extraction process is being performed. If an extraction process is being performed processing then proceeds to operation 230. In operation 230, the flags, illustrated in FIGs. 1A through 1D, are extracted based upon the field size determined in operation 210 and the specific data item desired. Thereafter, processing proceeds to operation 270 where the extracted information is stored in the destination register. Once stored processing proceeds to operation 280 where processing terminates. In an example embodiment shown in FIG. 6, the extraction process is further detailed as discussed ahead.

The paragraph beginning at page 9, line 18 is amended as follows:

If in operation 220 it is determined that an extraction process is not desired, then processing proceeds to operation 240. In operation 240 it is determined whether a combination process executed by the condition check module 175 for the arithmetic flags illustrated in FIGs. 1A through 1D is desired. If a combination process is not desired then processing proceeds to operation 280 where again processing terminates. However, if a combination process executed by the condition check module 175 is desired for the flags associated with several data items shown in FIGs. 1A through 1D, then processing proceeds to operation 250. In operation 250, the flags for each data item in the SIMD PSR register are extracted based on the field size determined in operation 210. Processing then proceeds to operation 260 where the extracted flags for each data item are combined based upon the function desired. Specific examples of combination functions for an AND operation and an OR operation are further detailed in the discussion of FIG. 4 and FIG. 5, respectively. Thereafter, processing proceeds to operation 270 where the results of the combined flags are stored in the destination register for access by the processor. Processing then terminates in operation 280.

The paragraph beginning at page 10, line 10 is amended as follows:

FIG. 4 is an of a flowchart of an AND function used in an example embodiment of the present invention and may be executed by the condition check module 175. Processing for this AND operation begins in operation 300 and immediately proceeds to operation 310. In

operation 310 it is determined whether the data field size is four bits (one nibble) in length. If the data field size is four bits in length then processing proceeds to operation 320. In operation 320, bits 31 through 28 of the destination register are set equal to bits 31 through 28 ~~anded~~ ANDed with bits 27 through 24 ~~anded~~ ANDed with bits 23 through 20 ~~anded~~ ANDed with bits 19 through 16 ~~anded~~ ANDed with bits 15 through 12 ~~anded~~ ANDed with bits 11 through 8 ~~anded~~ ANDed with the 7 through 4 and 3 through 0 of the SIMD PSR register. Thereafter, processing proceeds to operation [[320]] 330 where the remaining bits 27 through 0 of the destination register are set to zero. Processing then proceeds to operation 395 where processing terminates.

The paragraph beginning at page 10, line 22 is amended as follows:

Still referring to FIG. 4, if in operation 310 it is determined that a ~~four-bits~~ four-bit data field is not specified then processing proceeds to operation 340. In operation 340, it is determined whether an [[8 bit]] 8-bit (byte) data field is specified. If an 8 bit data field is specified in the SIMD data word, shown in FIG. 1B, then processing proceeds to operation 350. In operation 350, bits 31 through 24 of the destination register are set equal to bits 31 through 24 ~~anded~~ ANDed with bits 23 through 16 ~~anded~~ ANDed with bits 15 through 8 and bits 7 through 0 of the SIMD PSR register. Thereafter, processing proceeds to operation 360 where bits 23 through 0 of the destination register are set to zero. Processing then terminates in operation 395.

The paragraph beginning at page 11, line 7 is amended as follows:

Still referring to FIG. 4, if in operation 340 it is determined that an [[8 bit]] 8-bit data field is not specified, then processing proceeds to operation 370. In operation 370 it is determined whether a 16-bit (half word) data field is specified. If a 16-bit data field is specified, as shown in FIG. 1C, then processing proceeds to operation 380. In operation 380, bits 31 through 16 of the destination register are set equal to bits 31 through 16 ~~anded~~ ANDed with bits 15 through 0 of the SIMD PSR register. Thereafter, processing proceeds to operation 390 where bits 15 through 0 of the destination register are set to zero. Then, in operation 395, processing is terminated.

The paragraph beginning at page 11, line 15 is amended as follows:

FIG. 5 is an of a flowchart of an OR function used in an example embodiment of the present invention and may be executed by the condition check module 175. Processing for this OR operation begins in operation 400 and immediately proceeds to operation 410. In operation 410 it is determined whether the data field size is four bits (one nibble) in length. If the data field size is four bits in length then processing proceeds to operation 420. In operation 420, bits 31 through 28 of the destination register are set equal to bits 31 through 28 [[ORD]] ORed with bits 27 through 24 [[ORD]] ORed with bits 23 through 20 [[ORD]] ORed with bits 19 through 16 [[ORD]] ORed with bits 15 through 12 [[ORD]] ORed with bits 11 through 8 [[ORD]] ORed with the 7 through 4 [[ORD]] ORed with 3 through 0 of the SIMD PSR register. Thereafter, processing proceeds to operation [[420]] 430 where the remaining bits 27 through 0 of the destination register are set to zero. Processing then proceeds to operation 495 where processing terminates.

The paragraph beginning at page 12, line 3 is amended as follows:

Still referring to FIG. 5, if in operation 410 it is determined that a ~~four bits~~ four-bit data field is not specified, then processing proceeds to operation 440. In operation 440, it is determined whether an [[8 bit]] 8-bit (byte) data field is specified. If an [[8 bit]] 8-bit data field is specified in the SIMD data word shown in FIG. 1B, then processing proceeds to operation 450. In operation 450, bits 31 through 24 of the destination register are set equal to bits 31 through 24 [[ORD]] ORed with bits 23 through 16 [[ORD]] ORed with bits 15 through 8 [[ORD]] ORed with bits 7 through 0 of the SIMD PSR register. Thereafter, processing proceeds to operation 460 where bits 23 through 0 of the destination register are set to zero. Processing then terminates in operation 495.

The paragraph beginning at page 12, line 3 is amended as follows:

Still referring to FIG. 5, if in operation 440 it is determined that an [[8 bit]] 8-bit data field is not specified, then processing proceeds operation 470. In operation 470 it is determined whether a 16-bit (half word) data field is specified. If a 16-bit data field is specified, as shown in FIG. 1C, then processing proceeds to operation 480. In operation 480, bits 31 through 16 of the destination register are set equal to bits 31 through 16 [[ORD]] ORed with bits 15 through 0 of the SIMD PSR register. Thereafter, processing proceeds to operation 490 where bits 15 through 0 of the destination register are set to zero. Then in operation 495 processing is terminated.

The paragraph beginning at page 13, line 22 is amended as follows:

The benefit resulting from the present invention is that a simple, reliable, fast method and computer program is provided that will enable a SIMD capable processor of extracting and/or combining arithmetic flags associated with multiple data items that have been the subject of mathematical operations. This method and computer program is of such in nature that complex logic is not required thus saving space, power requirements, and heat generated by a processor. Further, this method and computer program allows a SIMD capable processor ~~of operating~~ to operate at peak efficiency due to the simplicity of the logic required.